intel®

ARTICLE
REPRINT

AR 345

Build a VLSI-based workstation for the Ethernet environment

In a shared-resources, distributed-system environment, you can design a compact computer system using the latest chips to keep network node component count low and operations at hard-disk speeds.

Michael Webb, *Intel Corp*

Early distributed minicomputer systems used a variety of dissimilar networking methods, services and expansion provisions. As μC-based workstations invade the business environment, the need arises for standardized local networking that can quickly transfer information and make optimum use of shared resources. You can build an Ethernet workstation that's relatively simple and inexpensive by configuring a network using the system, hardware and software considerations described in this article.

Choosing Ethernet as the network environment makes good sense. For example, Ethernet makes it possible to share large, high-speed, remote file facilities and thus minimize or even eliminate the need for disk storage at the workstation. Similarly, you can eliminate printers at most stations by sharing higher quality printers that provide automatic print spooling and such special features as graphics with text and electronic typesetting.

These remote high-quality services are possible because Ethernet permits rapid, 10M-bps data transmission with even a large number of network users, and sharing makes the services cost effective. This shared approach to overall system configuration allows you to build a compact, high-performance workstation that is optimal for a local-area-network (LAN) environment.

Although most major components used in this workstation are Intel parts, many are available from second sources. You can substitute other parts with some design adjustment. For example, you can use a high-speed 68000 CPU. Its system interface is more complex, and the total parts count would increase. In the design described, the total component count can be fewer than 75 ICs, even with 256k bytes of 64k-bit dynamic RAM (32 ICs).

A system overview

Fig 1 shows an Ethernet workstation configured without disk drives or a printer. The two JEDEC 28-pin EPROMs that reside on the system bus store bootstrap, diagnostic and utility programs. The bootstrap program locates the correct file server on power-up and downloads the operating system over the net. File accesses to remote file servers depend on Ethernet speed to keep performance at the level experienced when hard disks are dedicated to a single workstation. With large system memory, say, 256k bytes with 64k RAMs or 1M byte with 256k RAMs, large applications programs and data files can reside in the workstation once loaded over the network.

Of course, to connect to the network, the workstation needs an Ethernet interface. This formerly required a board of MSI devices and one or more DMA channels. Today, a dedicated LAN coprocessor combined with an Ethernet serial interface chip can provide the complete Ethernet interface. An 82586 LAN coprocessor, which

ORDER NUMBER: 231207-001

implements the full Ethernet specification and is compatible with the IEEE 802.3 LAN standard, provides buffer management both concurrently and in real time so that you can take full advantage of Ethernet performance.

Using a dual-port memory system permits the various system processors to share as much as 1M byte of 150-nsec dynamic RAM, which runs at 8 MHz, without any wait states. One memory port is tied to the system bus, while the other port is tied to a display system through a multiplexed bus. Using this dual-bus arrangement offloads the system bus from such tasks as screen refresh.

The dual-channel communications controller has two serial I/O channels, each of which can be configured for a different protocol; one of its ports is used for the workstation's keyboard, and the other channel supports a local modem interface. The keyboard contains a battery-backed CMOS single-chip microcontroller (80C51) that controls the keyboard, has an interface for a mouse or digitizing pad and maintains semipermanent system configuration and real-time clock information even when the system is powered down.

The logic needed to read and execute high-level Ethernet communications tasks is in the LAN coprocessor. It accesses the Ethernet through a serial interface chip and a transceiver (**Fig 2**). The interface between the serial interface chip and the transceiver requires a crystal clock and a few capacitors and resistors.

The hardware interface between the LAN coprocessor and CPU (**Fig 3**) is even more straightforward, requiring only an inverter and transistor. The LAN coprocessor, which has the same pin configuration as the CPU, resides on the multiplexed CPU bus. To gain control of this bus, the LAN coprocessor uses its Hold/Hold Acknowledge (HOLD/HLDA) lines, which are directly wired to the CPU. The LAN coprocessor and CPU share a set of octal noninverting address latches and transceivers, through which they access the system bus and communicate with system memory. All commands and status concerning the Ethernet link are exchanged between the two through this memory.

The dual-port controller, through which system memory is accessed (**Fig 4**), gives priority to Port A, the port shared by the CPU and LAN coprocessor, thereby minimizing latency when receiving an Ethernet packet. This arrangement allows time for the

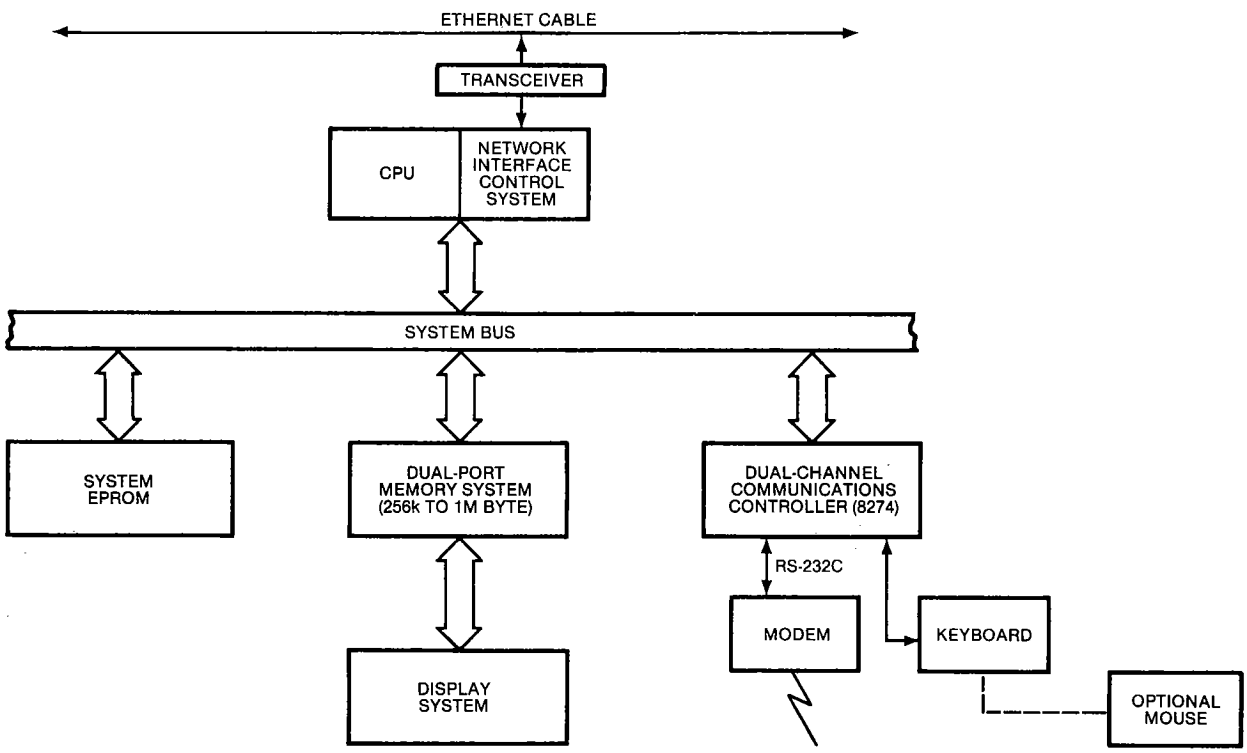


Fig 1—Consisting of four main subsections—*a network-interface control system, CPU, dual-port memory system and display controller—this Ethernet workstation relies on VLSI parts for high-speed performance. EPROM memory contains the bootstrap program that gets the system onto the net at power-up, while the dual-channel controller provides a keyboard interface and modem communication.*

workstation to receive a maximum-size Ethernet packet, while the text coprocessor, which resides on Port B of the memory controller, simultaneously fills a display buffer. By using a dual-port memory controller, you also shift the overhead incurred in filling a display buffer away from the main system bus, freeing it for other application tasks, such as communicating via the modem.

Because high-quality, low-cost printing is to be a network resource, your workstation should let you generate documents with different type fonts and graphics. A text display of 25 lines×132 characters and proportional-spacing capability is suitable for most business applications. Integrating a text coprocessor into the display system (**Fig 5**) makes sense because it can generate the proportionally spaced text using an LSI video interface component. Add a character generator and three latches for synchronization, and the display interface is complete.

Concurrent display of data from different files in separate areas of a CRT screen, called windows, has proven desirable in business applications and should be part of any modern workstation. In the past, hardware support for multiple windows was available only in $10,000-and-up workstations. A text coprocessor with on-chip DMA simplifies list-based manipulation of multiple overlaid text windows, as used with the Xerox Star and Apple Lisa workstations.

Looking at bandwidth realities

Considering the memory-intensive nature of this design, sufficient bandwidth in the dual-port memory system is a key factor. There must be sufficient bandwidth to support the data-rate requirements for display refresh. At the same time, the LAN coprocessor must be able to access memory frequently enough during packet reception or transmission to avoid overrun or underrun in its on-chip FIFO buffers.

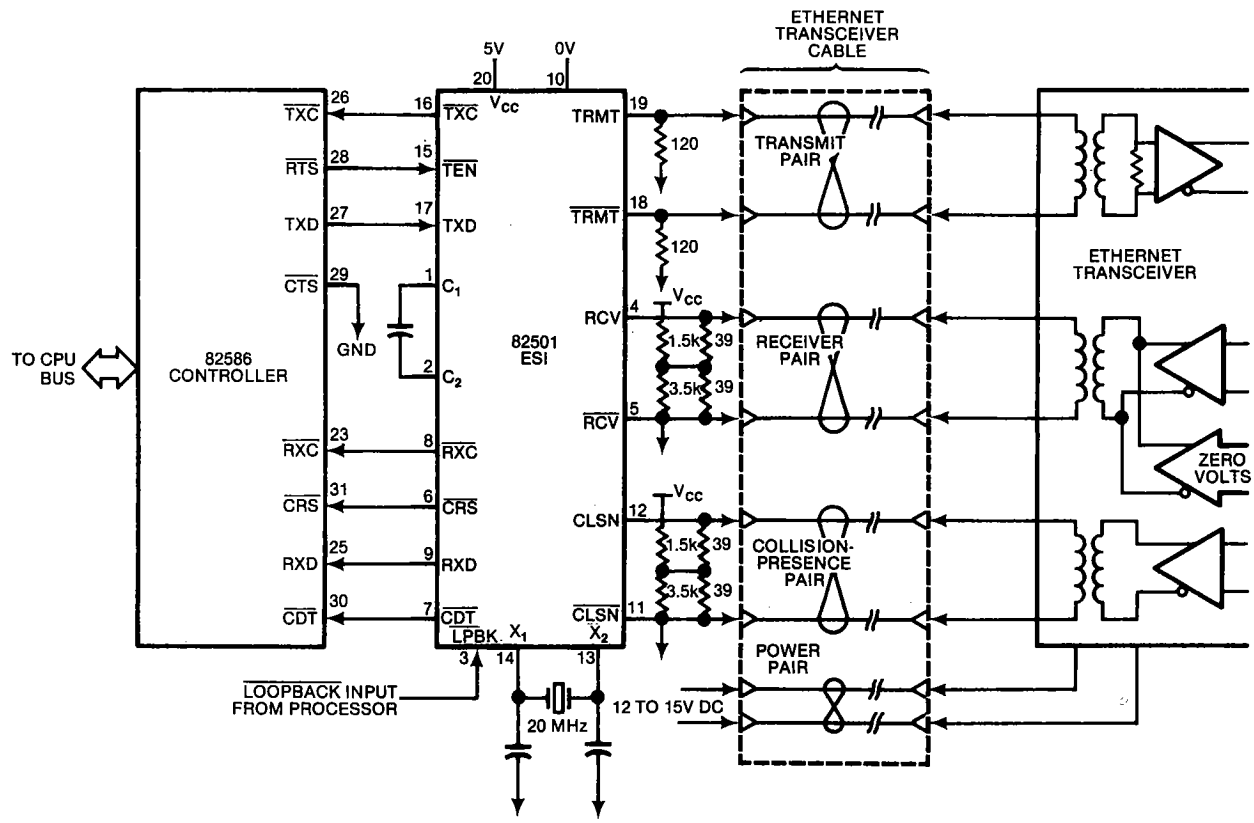


Fig 2—A serial interface chip (82501) provides a path *from LAN controller to cable transceiver. Its six outputs correspond with the six major signal lines of the 8-wire Ethernet cable. The remaining two lines are for dc power and return.*

An Ethernet workstation should support windows

To refresh a video display, you must fill the row buffer once for each character row displayed. If you use a display with 14 scan lines per character row, you can generate characters in a 9×14-character cell, which provides good character definition. Assuming that it takes 49 μsec to write a line (this timing depends on monitor selection), it would take 686 μsec to display a single row of characters.

If the text coprocessor must wait during every memory access for the LAN coprocessor or CPU to complete a cycle, and Port A of the dual-port controller has priority, then the worst-case memory-access time would be approximately 800 nsec per word. With 132 words per row and 20% overhead for string and pointer manipulation, 126.72 μsec are needed to fill the next row buffer. This is easily within the display refresh requirements, because 686 μsec are required to display a row. Because the text processor has dual row buffers, it makes no bus accesses 81% of the time and still provides continuous screen refresh.

When receiving packets from the Ethernet, the LAN coprocessor takes charge of the bus for the time required to store two maximum-size packets (1518 bytes each) that can arrive with a minimum interframe spacing of 9.6 μsec (the worst-case situation defined in the Ethernet specification). The LAN processor uses the 9.6-μsec gap between frames to set up pointers to the next free buffer. Without an intelligent controller, this time would be insufficient to prepare for the next frame.

Data arrives over the Ethernet at 800 nsec per byte. At 800 nsec per byte multiplied by 3036 (2×1518) bytes, it takes 2.4288 msec to receive these frames. During this time, for 126.72 μsec of every 686 μsec, the text coprocessor is contending with the LAN coprocessor for data to load its row buffers. Under worst-case conditions, the LAN processor can write a word every 800 nsec during this contention interval; without contention, it can write a word every 400 to 450 nsec. Either way, there is plenty of time to store data coming in at 1600 nsec per word.

In fact, the LAN processor in this system can continuously store incoming packets with the minimum interframe spacing as long as receive-buffer space is

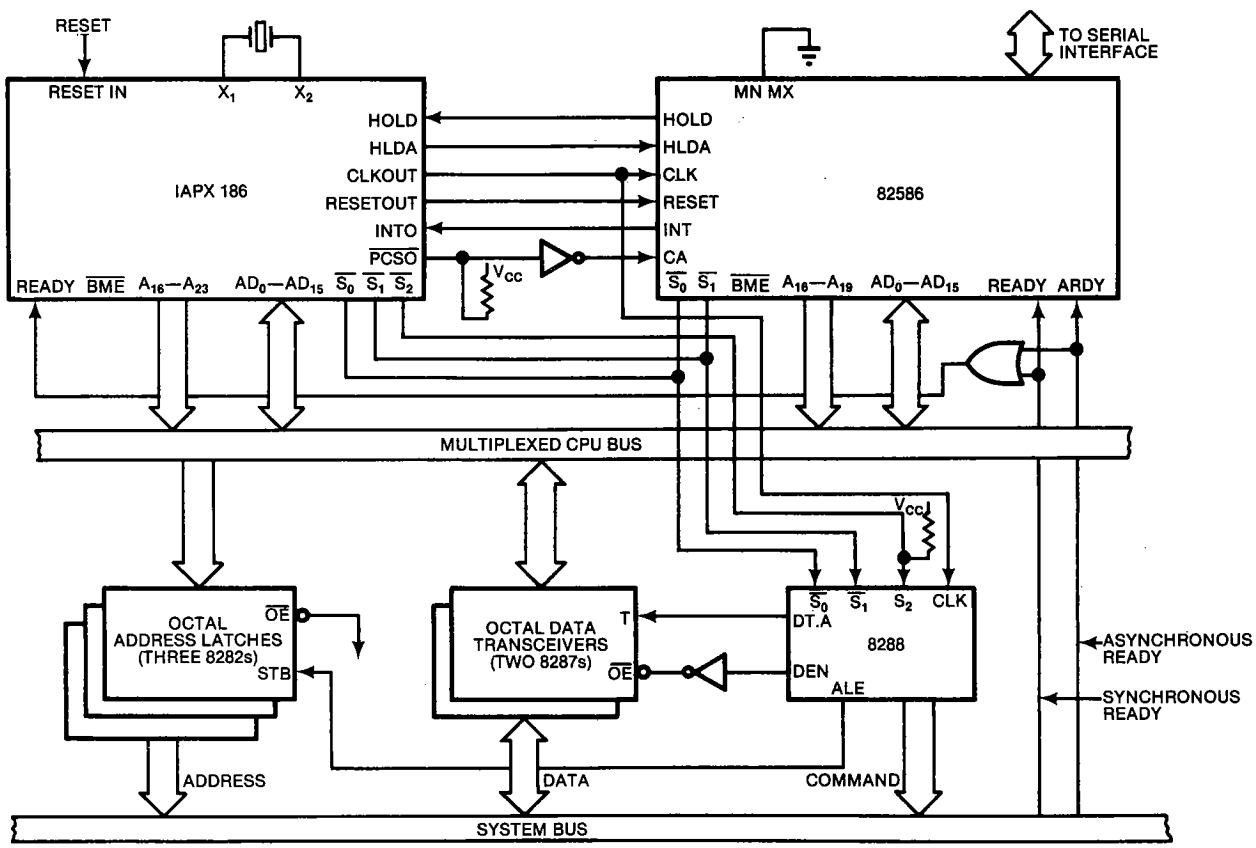


Fig 3—For the core of an Ethernet workstation, *an 80186 μP is teamed with an 82586 LAN controller to provide computational power and network communication, respectively. The units share the local CPU bus. No random logic is needed to interface them because they have identical interface structures and timing requirements.*

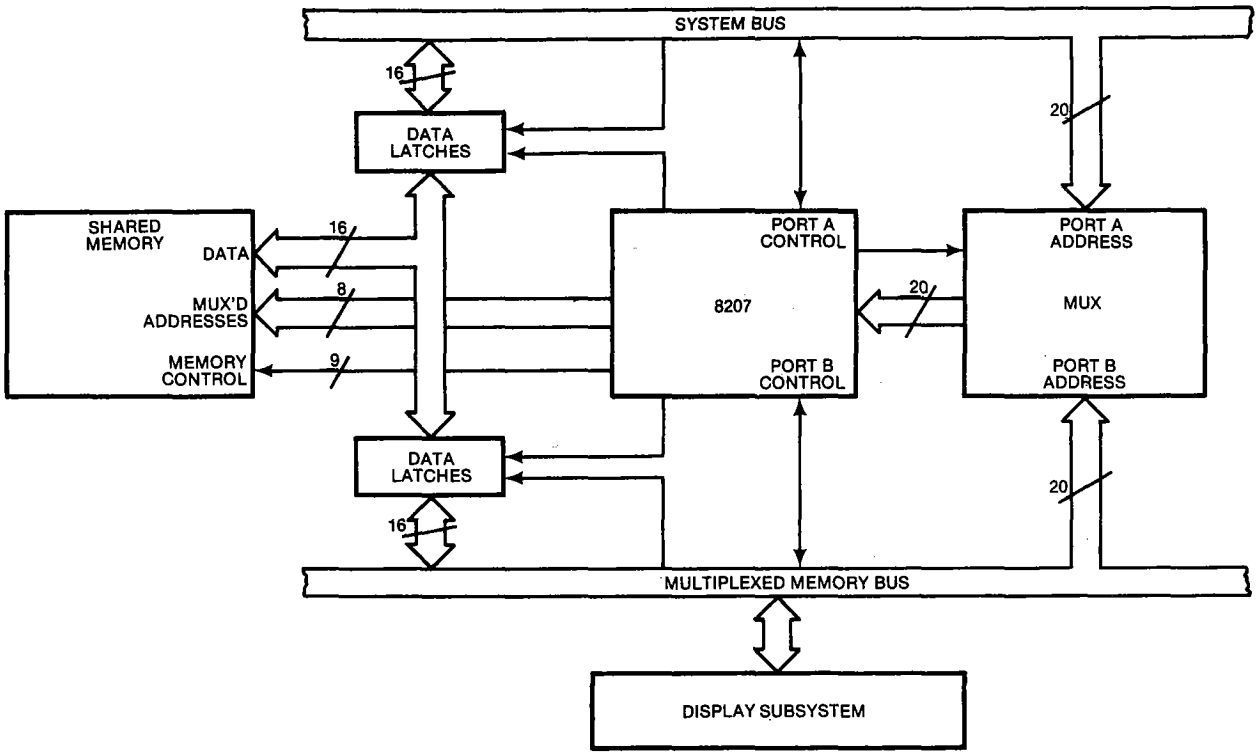
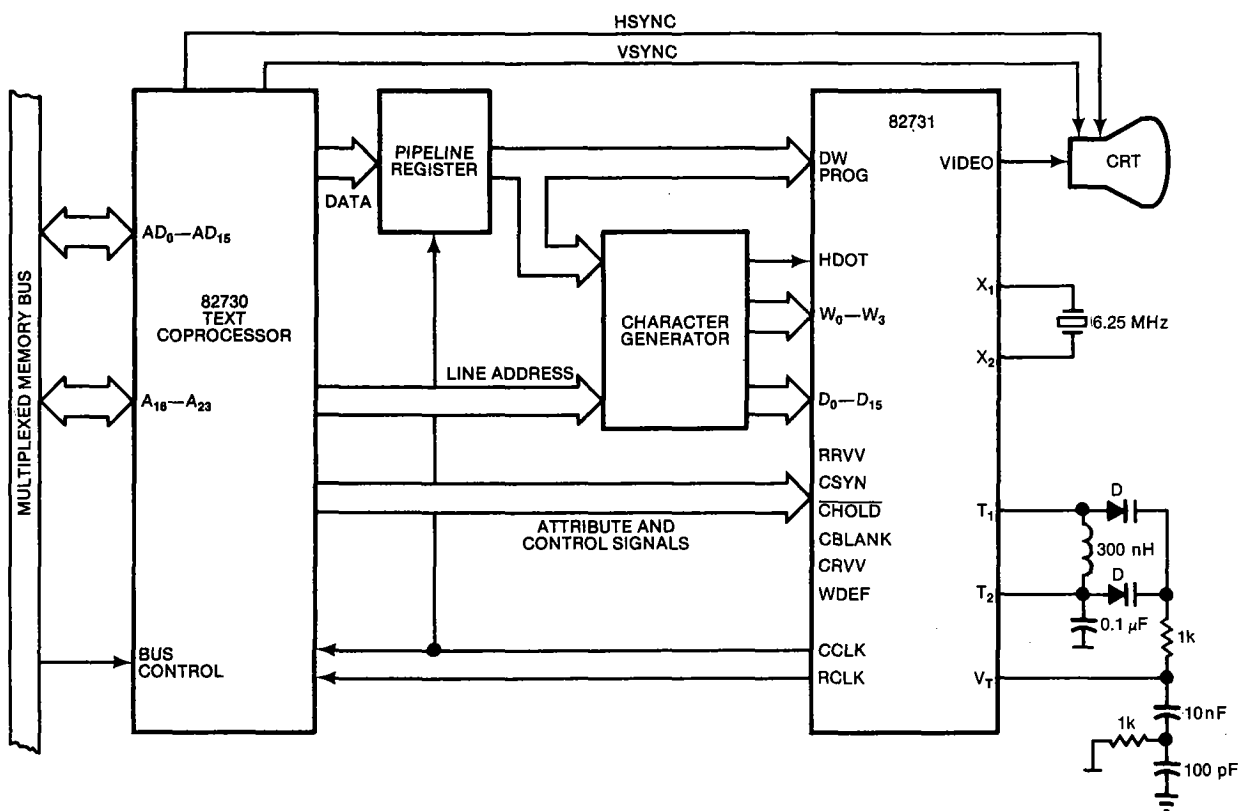


Fig 4—The memory system has two ports *tied to the system bus and the memory bus, which communicate with the display. Offloading display-memory accesses keeps the system bus open for Ethernet operations and user applications.*



D = BB505 VARACTOR DIODE OR EQUIVALENT

Fig 5—For the CRT subsystem, *the 82730 and 82731 convert bytes from the workstation bus to characters on a 25-line×132-character CRT screen. The text coprocessor also handles multiple windows.*

available in memory. At the same time, ⅓ to ½ of the bus bandwidth is still available for the CPU to continue program execution. In previous systems, program execution virtually stopped while high-bandwidth peripherals are using the bus. Because both peripherals are coprocessors, they run asynchronously and concurrently with other system activity.

With plenty of leftover bandwidth, the two DMA channels on the CPU can be used to add efficiency and performance to the dual-channel communications controller. Data from the keyboard-input buffer is transferred to the CPU via DMA, and the other DMA channel makes possible a 64k-baud synchronous or HDLC modem link on the communications controller's other port. Baud-rate timing for the two channels is generated using two of the CPU's three on-chip timers.

The CPU also directly generates chip selects, channel attentions and wait states for the system peripherals. The CPU's on-chip interrupt controller services interrupt inputs from the LAN coprocessor, text coprocessor, communications controller and other peripherals.

The software relationship

Most of the system's hardware relationships are easily grasped, but a firm understanding of the software architecture is essential to building an optimum workstation. **Fig 6** shows how the LAN coprocessor interacts with the system from a software point of view. Receive and Command units are software constructs rather than physical segments of the LAN coprocessor; in reality, the same hardware performs both functions.

As previously noted, all communications between the CPU and the LAN coprocessor occur in system memory. The CPU builds a command block, stores it in memory, updates the command-block list and then activates channel attention to get the LAN coprocessor to look at the command-block list for one or more new commands. If requested by the command block, the LAN coprocessor interrupts the CPU on completion of one or more commands.

The focal point for all interaction is the system control block (**Fig 7**). This data structure contains chip status, pointers to the command-block list and receive-frame areas, and universal statistics on faults such as CRC errors and alignment errors.

Both the command-block list and receive-frame area use the same concept, or model, to manage data buffers for either transmission or reception. The buffer management model employs one or more arbitrarily sized buffers to construct each data frame. Pointers control and access the buffers, and linked lists manipulate them.

This model offers distinct advantages over more primitive approaches. Allowing the physical buffers to be arbitrary sizes gives the system designer maximum flexibility in selecting the buffer size and in allocation methods. Because you can locate the memory for these buffers anywhere in the 16M-byte address space, this buffer management support simplifies the task for the

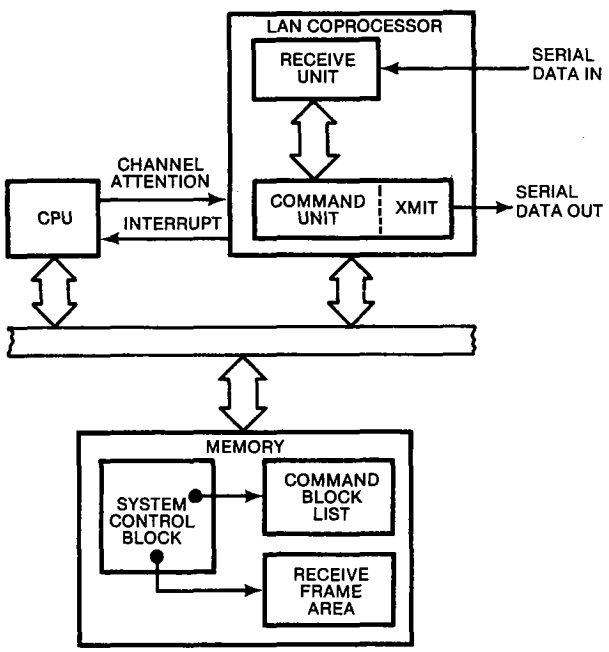


Fig 6—Seen from a software perspective, *the LAN coprocessor looks like Receive and Command units. When receiving data packets from Ethernet, the coprocessor converts them from serial form and places them in frame area locations it manages within the memory system. The CPU directs the coprocessor using handshake lines and messages left in the shared-memory system's command-block list.*

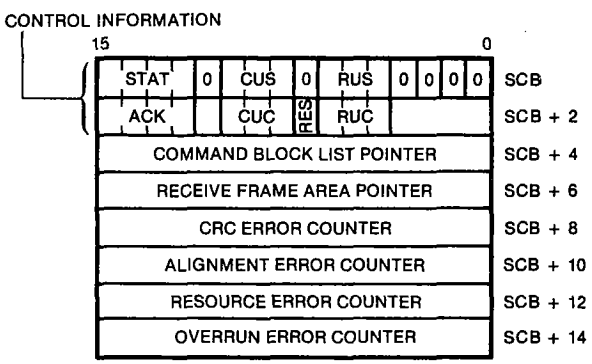


Fig 7—A special memory block, the system control block, *serves as a bulletin board whereby the CPU and LAN controller communicate. The system control block holds control and status information pointers to the command-block list, where the CPU stores instructions for the LAN coprocessor, and the receive-frame area, where data from the Ethernet is stored by the LAN coprocessor. The block also contains error information of interest to the CPU.*

operating system's dynamic memory-allocation scheme. Communication buffers for the LAN coprocessor are dynamic by definition.

A buffer needn't be the size of the largest frame ever expected; it can be any convenient size. If a frame larger than the selected size arrives, the LAN coprocessor automatically allocates as many buffers as necessary to contain the frame, and updates the pointers and links to indicate where the frame starts and which buffers are occupied by the frame.

This flexible buffer size avoids the waste of large, dedicated buffers for receive frames when most of the frames actually received are much smaller than the maximum size (ie, are control frames rather than data frames). Also, this automatic buffer chaining of receive data increases communications performance and efficiency; even if several frames arrive before the CPU is free to examine incoming data, the workstation seldom, if ever, misses a frame addressed to it.

The effectiveness of this buffer management scheme is perhaps best understood by examining what happens when a packet comes in from Ethernet. The LAN coprocessor's Receive section handles all frame reception activities. It manages a pool of memory space—the receive-frame area (**Fig 8**)—using the receive-frame and free-frame lists.

Within each list are receive-frame descriptors (RFDs) that contain status data and pointers. The RFDs in the receive-frame area point both to the first buffer that has been filled with received data, and to the first RFD of the free-frame area. The CPU can organize the pointers in linear, random or circular fashion. Once a pointer structure is adopted, the LAN coprocessor allocates buffers and maintains the proper linkage automatically.

When the LAN coprocessor begins receiving a frame, it uses the first RFD in the free-frame list to hold status and information concerning the frame, and then allocates and links in as many free buffers as necessary to contain the frame data. The linking

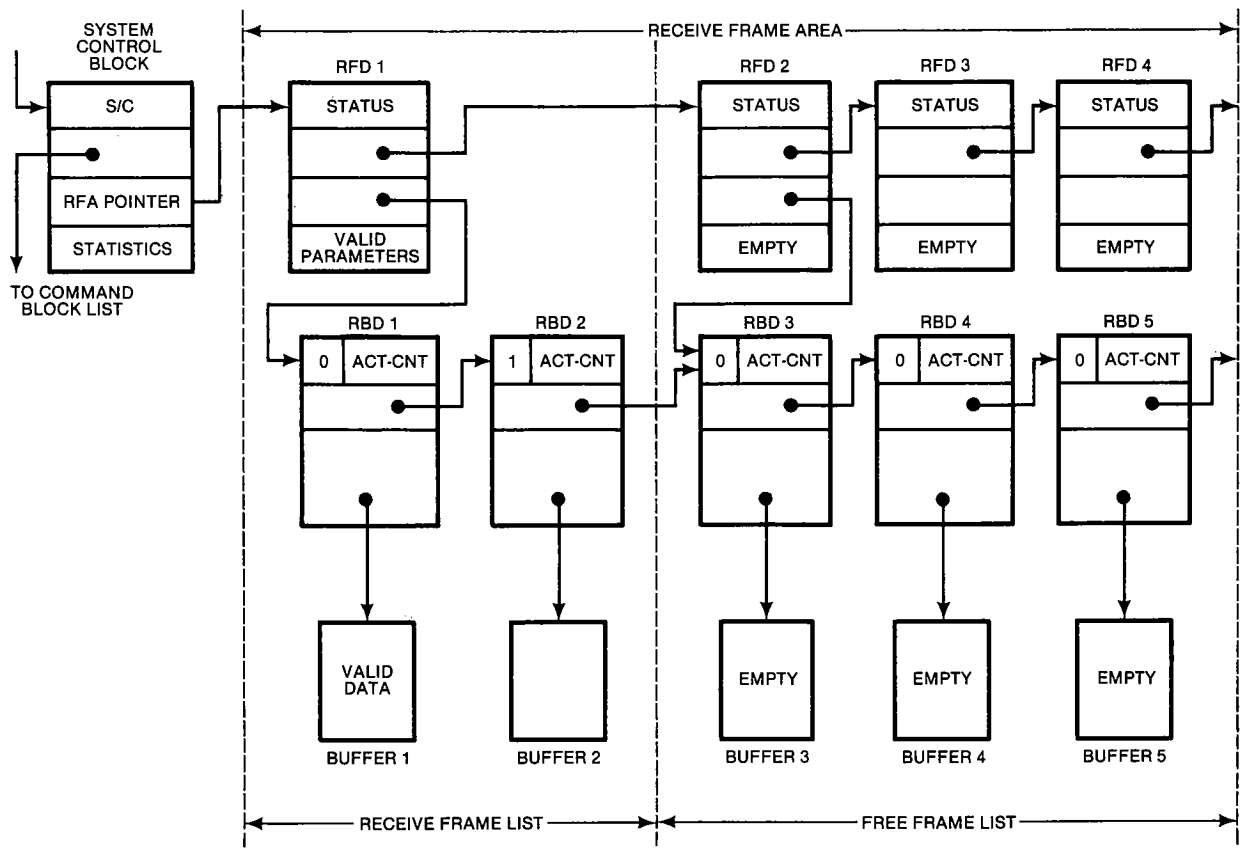


Fig 8—Received packets *are stored in a receive-frame area consisting of two linked lists—the receive-frame list and free-frame list. The LAN coprocessor pulls into the receive-frame list as many buffers of any size as needed to store an incoming packet. Buffers used are pointed to by receive-frame descriptor (RFD) and receive-buffer descriptor (RBD) blocks. The variable-size message block saves memory, with packets stored according to actual size rather than maximum size.*

process is accomplished using a receive-buffer descriptor (RBD) to point to the next buffer containing contiguous data.

Once frame data is complete, the LAN coprocessor writes the frame status into the associated RFD and the actual count of valid data into each RBD used by the frame. It then flags the last RBD used to contain the frame and updates the first RFD on the free list to point to the first free RBD. All this is done in time to catch a second frame sent to the workstation address, if one is transmitted immediately following completion of the previous frame using Ethernet's 9.6-μsec minimum frame spacing.

In effect, the LAN coprocessor can receive continuous data packets from the network as long as buffer space remains available. This is achieved by dedicating a complete microcoded machine in the LAN coprocessor to generating buffer management primitives, and giving this machine DMA control to speed its bus requests. No DMA setup or control is needed from the CPU, reducing overhead and simplifying the system.

Data transmission is accomplished in a similar fashion. To transmit a frame over the Ethernet via the LAN coprocessor, the CPU constructs a command block (**Fig 9**). Included in this command block is a pointer to a buffer descriptor, which points to one or more buffers containing the data to transmit.

If more than one buffer is used, the LAN coprocessor automatically links the buffers together as it transmits the frame. In addition, the LAN coprocessor automatically inserts the frame preamble, source and destination addresses, type field, and CRC during the transmission process. The CPU can choose to be interrupted following the transmission of one frame, or it can link together several transmission requests and be interrupted following the final frame transmission. **EDN**

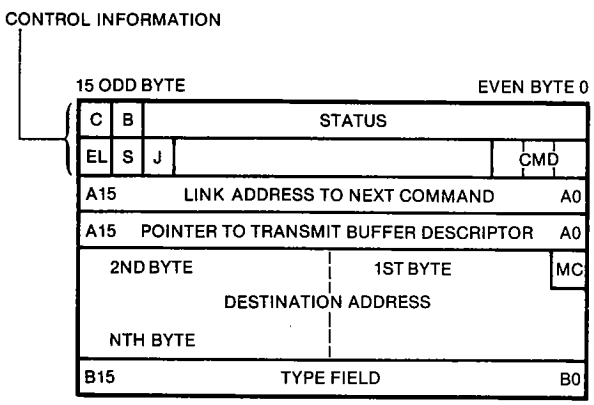
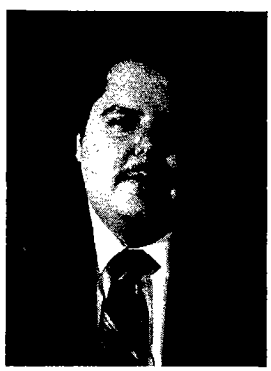


Fig 9—The system control block points at *the command block containing instructions from CPU to LAN controller concerning a transmission. These include control information, a link to the next command, a pointer to a transmit-buffer descriptor, the packet's destination address and the type of field it contains. The command area is organized similarly to the receive-frame area.*

References

1. Metcalf, R M, and Boggs, D R, "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the ACM,* Vol 19, No. 7, July 1976, pp 395-403.
2. Shoch, J F, and Hupp, J A, "Performance of an Ethernet Local Network: A Preliminary Report," *Proceedings of the Local Area Communications Network Symposium,* May 1979, pp 113-124.
3. Tobagi, F A, "Message-based priority functions in Multiaccess/Broadcast Communications Systems with carrier sense capability," *Stanford University Electronic Labs Technical Report,* No. 181, October 1979.
4. Digital Equipment Corp, Intel Corp, and Xerox Corp, "The Ethernet Specification," Ver 2.0, November 1982.

Author's biography



Michael Webb is a regional applications specialist for Intel Corp (Dallas, TX), where his primary job is to help customers define architectures and designs for distributed systems, such as the Ethernet system described in this article. No stranger to Ethernet, Mike previously worked at Xerox Corp in the Office Products Div where he was engineering manager for Ethernet communications software. Mike's hobbies include devising game software and chess.

INTEL CORPORATION, 3065 Bowers Ave.
Santa Clara, CA 95051; Tel. (408) 987-8080

INTEL INTERNATIONAL CORPORATION Ltd.
Swindon, United Kingdom; Tel. (0793) 488 388

INTEL JAPAN k.k., Ibaraki-ken; Tel. 029747-8511

Printed in U.S.A./D-4043/10K/9-84/SCP/PM
peripherals